Data Replication

Database Replication

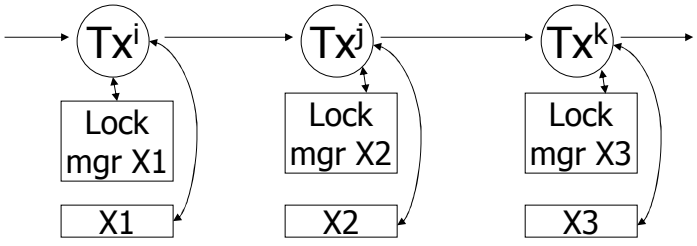
- Replication is a common strategy in data management
  - RAID (Redundant Array of Independent Disks) technology, Mirror sites for web pages, backup mechanisms (1-safe, hot/cold standby)
  - Our focus will be on replicated databases but many of the ideas we discuss will apply to other environments as well
- If you're not careful, replication can lead to
  - worse performance - updates must be applied to all replicas and synchronized
  - worse availability - some algorithms require multiple replicas to be operational for any of them to be used

Why Replicate?

- Performance
  - Local accesses are fast, remote accesses are slow.
  - If everything is local, then all accesses should be fast.
- Fault Tolerance (Availability)
  - If a site fails, the data it contains becomes unavailable.
  - Keeping several copies of the data at different sites, single site failures should not affect the overall availability
- Scalability
  - In numbers and geographic area
- Application Type
  - Databases have always tried to separate queries from updates to avoid interference, resulting in two types of application: OLTP and OLAP depending on whether they are update or read intensive

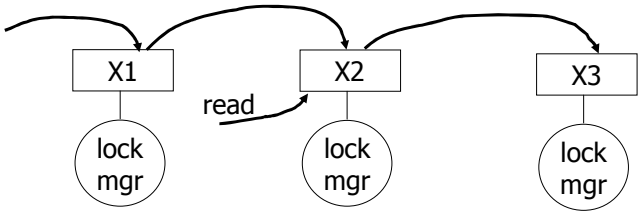
Basic Solution

- Treat each copy as an independent data item

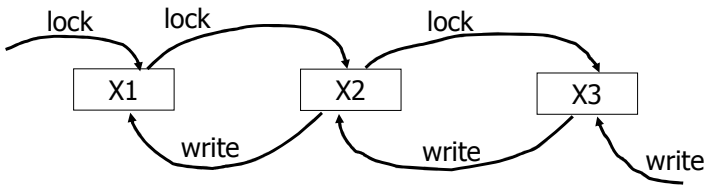


Object X has copies X1, X2, X3

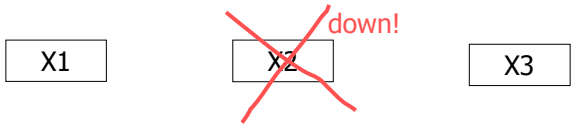
- Read(X):
  - get shared X1 lock
  - get shared X2 lock
  - get shared X3 lock
  - read one of X1, X2, X3
  - at end of transaction, release X1, X2, X3 locks



- Write(X):
  - get exclusive X1 lock
  - get exclusive X2 lock
  - get exclusive X3 lock
  - write new value into X1, X2, X3
  - at end of transaction, release X1, X2, X3 locks



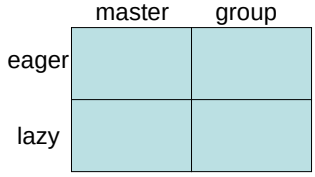
- Correctness OK
  - 2PL ⇒ serializability
- Problem: Low availability and concurrency



➥ cannot access X!

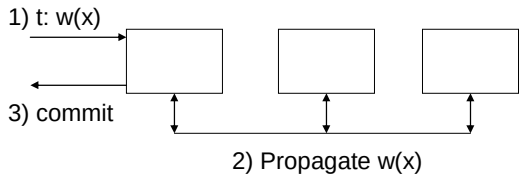
How to replicate data?

- Two basic parameters to select when designing a replication strategy: *when* and *where*.
- Depending on when the updates are propagated:
  - Synchronous (eager)
  - Asynchronous (lazy)
- Depending on where the updates can take place:
  - Primary Copy (master)
  - Update Everywhere (group)



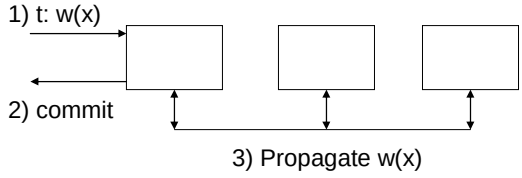
When to propagate update?

- Synchronous (Eager)
  - Within the boundaries of the transaction
  - Propagates any changes to the data immediately to all existing copies
    - Can be deferred as long as within transaction boundaries
  - ACID properties apply to all copies
  - Transactions terminate usually with 2PC



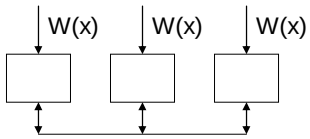
When to propagate update? (Cont)

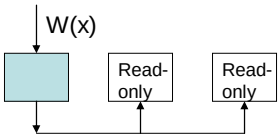
- Asynchronous (Lazy)
  - Updating transaction is first executed on the local copy. Then the changes (*refreshed transactions*) are propagated to all other copies (after transaction commits)
  - While the propagation takes place, the copies are inconsistent (data may have different values)
  - The time the copies are inconsistent is an adjustable parameter which is application dependent



Where to submit updates?

- Update Everywhere (Distributed)
  - Update transactions can be executed at any site, i.e., any site which owns a copy can update the value of the data item
- Primary Copy (Centralized)
  - There is only one copy which can be updated (the master)
  - All others (secondary copies) are read-only, and are updated reflecting the changes to the master
  - Also called master/slave





Forms of replication

Synchronous
- Advantages
  - No inconsistencies (identical copies)
  - Reading the local copy yields the most up-to-date value
  - Changes are atomic
- Disadvantages
  - A transaction has to update all sites (longer execution time, worse response time)
  - Lower availability

Asynchronous
- Advantages
  - Transaction is always local (good response time)
- Disadvantages
  - Data inconsistencies
  - A local read does not always return the most up-to-date value
  - Changes to all copies are not guaranteed

Update Everywhere
- Advantages
  - Any site can run a transaction
  - Load is evenly distributed
- Disadvantages
  - Different copies may be updated at different sites concurrently
  - Copies need to be synchronized

Primary Copy
- Advantages
  - No inter-site synchronization is necessary (it takes place at the primary copy)
  - There is always one site which has all the updates
- Disadvantages
  - The load at the primary copy can be quite large
  - Reading the local copy may not yield the most up-to-date value

Replication Strategies

Synchronous/Primary Copy
- Advantages
  - Updates do not need to be coordinated
  - No inconsistencies
- Disadvantages
  - Longest response time
  - Only useful with few updates
  - Local copies can only be used for read

Asynchronous/Primary Copy
- Advantages
  - No coordination necessary
  - Short response time
- Disadvantages
  - Local copies are not up-to-date
  - Inconsistencies

Synchronous/Update Everywhere
- Advantages
  - No inconsistencies
  - Elegant (symmetrical solution)
- Disadvantages
  - Long response time
  - Updates need to be coordinated

Asynchronous/Update Everywhere
- Advantages
  - No centralized coordination
  - Shortest response time
- Disadvantages
  - Inconsistencies
  - Updates can be lost (reconciliation)

Replication Strategies

	Primary Copy	Update Everywhere
Eager	Globally correct Remote writes (Too expensive, Useful?)	Globally correct Local writes (does not scale)
Lazy	Inconsistent reads (Feasible)	Inconsistent reads Reconciliation (Feasible in some applications

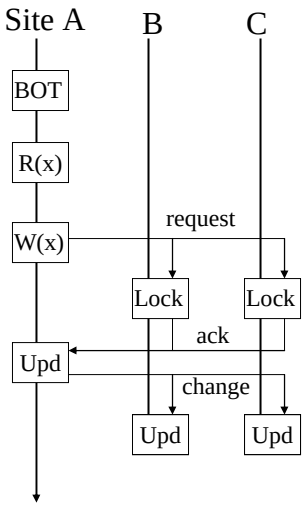
Managing Replication

- When data is replicated, we still need to guarantee:
  - Isolation
    - Transaction do not interfere with each other
    - Can be guaranteed by a concurrency control protocol
    - 2 Phase Locking (2PL) guarantees serializability
  - Atomicity
    - A transaction must commit at all sites or none at all
    - Can be guaranteed by using 2 Phase Commit
    - The challenge is how to make sure the serialization orders are the same at all sites, i.e., make sure that all sites do the same things in the same order (otherwise the copies would be inconsistent)
      - Replication protocol is needed
      - Specifies how different sites must be coordinated in order to provide a concrete set of guarantees
      - Depend on the replication strategy

Synchronous-Update Everywhere

READ ONE WRITE ALL (ROWA)

- Each site uses 2 PL
- Read operations are performed locally
- Write operations are performed at all sites (using a distributed locking protocol)
  - Write locally
  - Request other locks, obtain locks, receive acknowledgement
- This protocol guarantees that every site will behave as if there were only one database. The execution is serializable (correct) and all reads access the latest version
- Transaction is committed using 2PC



Synchronous-Update Everywhere

READ ONE WRITE ALL (ROWA)

- This simple protocol illustrates the main idea behind replication, but it needs to be extended in order to cope with realistic environments:
  - Sites fail, which reduces availability (if a site fails, no copy can be written)
  - Sites eventually have to recover (a recently recovered site may not have the latest updates)

Dealing with Site Failures (No Communication Failures)

WRITE ALL AVAILABLE (ROWAA)

- READ – read any copy, if time-out, read another copy
- WRITE – send WRITE(x) to all copies. If one site rejects the operation, then abort; otherwise, all sites not responding are “missing writes”
- VALIDATION – to commit a transaction
  - Check that all sites in “missing writes” are still down. If not, then abort the transaction
  - Check that all sites that were available are still available. If some do not respond, then abort.
- Recovery: Get missed updates from active node
- Need to poll nodes to know which nodes are active/down

Dealing with Comm. Failures

To tolerate site and communication failures

- QUORUM PROTOCOL
- Quorums are sets of sites formed in such a way so as to be able to determine that it will have a non-empty intersection with other quorums
  - Simple majorities (site quorums)
  - Weighted majorities (quorum consensus)

Weighted Quorums

Quorum Consensus Protocol

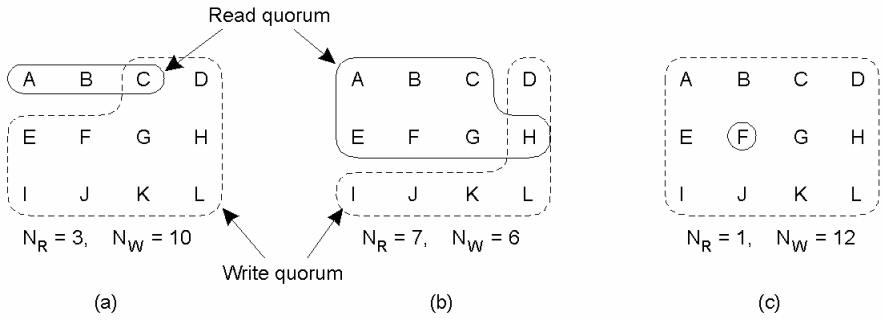
- Each copy has a weight assigned to it
- Total weight of all copies is N
- Let RT and WT be read and write thresholds, respectively, such that
  - 2 WT > N
  - RT + WT > N
- A read quorum is a set of copies such that their total weight is greater or equal to RT
- A write quorum is a set of copies such that their total weight is greater or equal to WT

Weighted Quorums

- Each copy has a version number
- READ – contact sites until a read quorum is formed, then read the copy with the highest version number
- WRITE – contact sites until a write quorum is formed; get the version number of the copy with the highest version number (k); write to all sites in the quorum adding the new version number (k+1)

Recovery is for FREE! But reading is no longer local and it is not possible to change the system dynamically (the copies must be known in advance)

Quorum Example



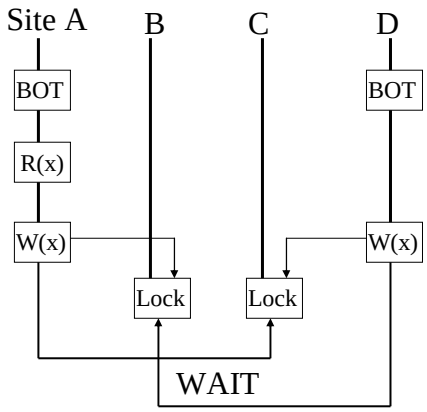
Three examples of the voting algorithm:

a) A correct choice of read and write set

b) A choice that may lead to write-write conflicts

c) ROWA

The Deadlock Problem



Synchronous-Update Everywhere

Advantages

- No inconsistencies
- Elegant (symmetrical solution)
- High fault tolerance

Disadvantages

- Very high number of messages involved
- Transaction response time is very long
- System will not scale because of deadlocks (as the number of nodes increases, probability of getting into a deadlock gets too high)

Synchronous-Primary Copy

All write transactions are sent to the Primary Copy site. Read-only transactions can be processed at Secondary Copy site.

Primary Copy

- READ – read locally
- WRITE – write locally, multicast write to other replicas in FIFO order
- COMMIT – 2PC
- ABORT – abort and inform other sites to abort

Secondary Copy

- READ – read locally
- WRITE from primary
- WRITE from client – refuse
- COMMIT request from read-only: Commit locally
- Participant of 2PC

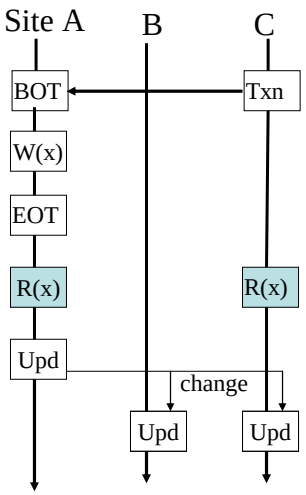
Deadlocks: Secondary copies should abort reading transactions

Synchronous-Primary Copy

- Advantages
  - Updates do not need to be coordinated
  - No inconsistencies
- Disadvantages
  - Longest response time
  - Only useful with few updates (primary copy is bottleneck)
  - Local copies are almost useless
  - Not used in practice

Asynchronous-Primary Copy

- Update transactions are executed at the primary copy site
- Read transactions are executed locally
- After the transaction is executed, the changes are propagated to all other sites
- Locally, the primary copy site uses 2PL
- In this scenario, there is no atomic commitment problem (the other sites are not updated until later)



Asynchronous-Primary Copy

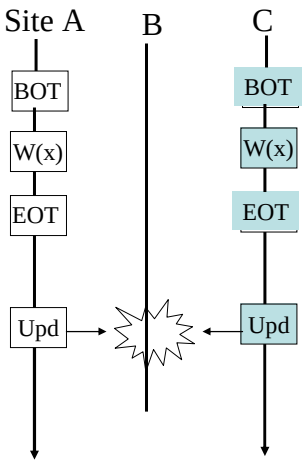
- Advantages
  - No coordination necessary
  - Short response time (transaction is local)
- Disadvantages
  - Local copies are not up-to-date (a local read will not always include the updates made at the local copy)
  - Inconsistencies (different sites have different values of the same data item)
  - Fault tolerance limited (almost same as no replication)

Asynchronous-Primary Copy

- Collect updates at primary using triggers or the log
- Triggers (Oracle8, Rdb, SQL Server, DB2, …)
  - On every update at primary, a trigger fires to store the update in the update propagation table.
- Post-process the log to generate update propagations
  (SQL Server, DB2, Tandem Non-Stop SQL)
  - Off-line, so saves trigger and triggered update overhead, though R/W log synchronization also has a cost
  - Requires admin (what if log reader fails?)

Asynchronous-Update Everywhere

- All transactions are executed locally (i.e., read, write, commit locally)
- After the transaction is executed, the changes are propagated to all other sites
  - Detect conflicts
  - Install changes
- Locally, a site uses 2PL
- In this scenario, there is no atomic commitment problem (other sites are updated later)
- However, unlike with primary copy, updates need to be coordinated



Asynchronous-Update Everywhere

- What does it mean to commit a transaction locally? There is no guarantee that a committed transaction will be valid (it may be eliminated if “the other value” wins)
- Such problems can be solved using pre-arranged patterns
  - Latest update win (new updates perferred over old ones)
  - Site priority (preference to updates from headquarters)
  - Largest value (the larger transaction is preferred)
- Or using ad-hoc decision making procedures
  - Identify the changes and try to combine them
  - Analyze the transactions and eliminate the non-important ones
  - Implement your own priority schemes

Asynchronous-Update Everywhere

- Advantages
  - No centralized coordination
  - Shortest response time
  - High fault tolerance
- Disadvantages
  - Inconsistencies
  - Updates can be lost (reconciliation is a tough problem – typically solved manually)

Summary

- Replication is used for performance and fault tolerance purposes
- There are four possible strategies to implement replication solutions
- We have seen the different technical issues involved with each replication strategy
- Each strategy has well-defined problems (deadlocks, reconciliation, consistency)
- There is a trade-off between correctness (data consistency) and performance (throughput and response time)